Disciplina: Projeto e Otimização de Algoritmos
Professor: Rafael Scopel

Revisão P1

Sugestão de roteiro de estudo para P1:

- Revisar os slides de aula;
- Executar e analisar os códigos java vistos em aula;
- Ler os capítulos 4,5 e 6 do livro: KLEINBERG, J.; TARDOS, E. Algorithm design. Addison-Wesley, 2005.
- Ler os capítulos 4, 15 e 16 do livro: CORMEN, T. H. Algoritmos: teoria e prática. 3a ed., Rio de Janeiro: Elsevier-Campus, 2012.

1. Você é um consultor trabalhando para uma empresa de caminhões que faz muitos negócios transportando mercadorias entre São Paulo e Porto Alegre. O volume é tão alto que eles precisam mandar vários caminhões por dia entre as duas localidades. Caminhões possuem um limite de carga máximo W. Caixas de mercadorias chegam na estação de São Paulo uma por uma, e cada caixa i possui um peso w_i. A estação de caminhões é bem pequena e apenas um caminhão por vez pode ser carregado na estação. A política da empresa exige que as caixas sejam enviadas na ordem em que chegam. Atualmente, a empresa está utilizando o seguinte algoritmo greedy para carregamento dos caminhões:

    - Carregar os caminhões com as caixas por ordem de chegada;
    - Quando a próxima caixa não couber mais no caminhão, o caminhão é liberado para entrega.

    Os gestores da empresa estão em dúvida se eles podem estar utilizando caminhões demais, e pediram a sua opinião se o procedimento de carregamento pode ser melhorado. O pensamento dos gestores é o seguinte: "E se diminuíssemos o número de caminhões, despachando um caminhão que não está completamente carregado, e assim permitindo que os próximos sejam carregados de forma mais eficiente."

    Prove que, para um conjunto de caixas com pesos específicos, o algoritmo greedy em uso realmente minimiza o número de caminhões necessários. Sua prova deveria seguir o mesmo tipo de análise que utilizamos para o problema de Escalonamento de Tarefas. Ou seja, estabelecer que o algoritmo é ótimo identificando uma medida pela qual o algoritmo greedy está sempre a frente das outras soluções.

Resposta:

Assuma que n caixas chegam na ordem $b_1, b_2, \dots, b_n$. Cada caixa b_i tem um peso positivo w_i, e o peso máximo que cada caminhão consegue transportar é W. Para armazenar as caixas em N caminhões preservando a ordem, precisamos alocar cada caixa para cada um dos caminhões $1,2, \dots, N$ de forma que:

- Nenhum caminhão fica sobrecarregado: o peso total de todas as caixas dentro do caminhão é menor ou igual a W.
- A ordem de chegada das caixas é preservada: se a caixa b_i é enviada antes da caixa b_j (ou seja, a caixa b_i é carregada no caminhão x e b_j é carregada no caminhão y, onde $x < y$) então b_i chegou na empresa antes da caixa b_j, ou seja $i < j$.

Demonstraremos que o algoritmo greedy usa a menor quantidade de caminhões demonstrando que ele sempre "está a frente" de outras soluções.

Suponha que o algoritmo greedy carrega as caixas $b_1, b_2, \dots, b_j$, nos primeiros k caminhões e uma outra solução carrega $b_1, b_2, \dots, b_i$ nos primeiros k caminhões, então **afirmamos** que $i \leq j$, ou seja, a solução alternativa carrega uma quantidade menor ou igual a solução greedy.

Note que isso implica que o algoritmo greedy é ótimo, definindo k como o número de caminhões utilizados pelo algoritmo greedy.

Provaremos essa afirmação por indução em k:

Caso: $k = 1$

O caso de $k = 1$ é claro; o algoritmo greedy coloca o maior número de caixas possíveis no primeiro caminhão.

Caso: $k > 1$

Agora, assumindo que a solução funciona para $k - 1$: o algoritmo greedy carrega j' caixas nos primeiros $k - 1$ caminhões, e a outra solução carrega i' caixas, onde $i' \leq j'$.

Agora para o k-ésimo caminhão, a solução alternativa carrega $b_{i'+1}, \dots, b_i$. Portanto, como $i' \leq j'$, o algoritmo greedy é capaz de carregar no mínimo todas as caixas $b_{j'+1}, \dots, b_i$ no k-ésimo caminhão, e potencialmente poderia carregar mais.

Isso demostra que o algoritmo greedy é ótimo, pois está sempre a frente da solução alternativa.

2. Resolva as seguintes recorrências utilizando o método mestre:

- $T(n) = 7T\left(\frac{n}{49}\right) + 1$
- $T(n) = 11T\left(\frac{n}{3}\right) + n^4$
- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

> ***TEOREMA T.1 O método mestre***
> - **Caso 1: Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.**
> - **Caso 2: Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \log_2 n)$.**
> - **Caso 3: Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e todos os n suficientemente grandes, então $T(n) = \Theta(f(n))$.**

a)

$$T(n) = aT(n/b) + f(n) = 7T\left(\frac{n}{49}\right) + 1$$

onde $a = 7,\ b = 49,\ f(n) = 1$

Então temos que $n^{log_b a} = n^{log_{49} 7} = \sqrt{n} = \Theta(\sqrt{n})$

Caso 1:

Como $f(n) = O(n^{\log_b a - \epsilon})$, onde $\epsilon = 0.5$, pois $n^{\log_b a - \epsilon} = n^{log_{49} 7 - 0.5} = n^{0.5-0.5} = 1$.

Portanto, $n^{log_b a} > f(n)$, a solução é $T(n) = \Theta(\sqrt{n})$.

b)

$$T(n) = aT(n/b) + f(n) = 11T\left(\frac{n}{3}\right) + n^4$$

onde $a = 11,\ b = 3,\ f(n) = n^4$

Então temos que $n^{log_b a} = n^{log_3 11} = n^{\frac{\log 11}{\log 3}} = \Theta(n^{2.18})$

Caso 3:

Como $f(n) = O(n^{\log_b a + \epsilon})$, onde $\epsilon = 1.82$, pois $n^{\log_b a + \epsilon} = n^{log_3 11 + 1.82} = n^{2.18+1.82} = n^4$.

$af(n/b) \leq cf(n) \rightarrow 11(n/3)^4 \leq cn^4 \rightarrow \frac{11}{81}n^4 \leq cn^4$
onde $c = \frac{11}{81} < 1$.

Portanto, $n^{log_b a} < f(n)$, a solução é $T(n) = \Theta(n^4) = \Theta(f(n))$.

c)

$$T(n) = aT(n/b) + f(n) = 4T\left(\frac{n}{2}\right) + n^2$$

onde $a = 4,\ b = 2,\ f(n) = n^2$

Então temos que $n^{log_b a} = n^{log_2 4} = n^2 = \Theta(n^2)$

Caso 2:

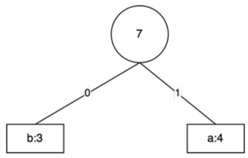
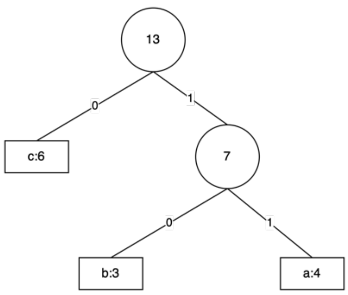
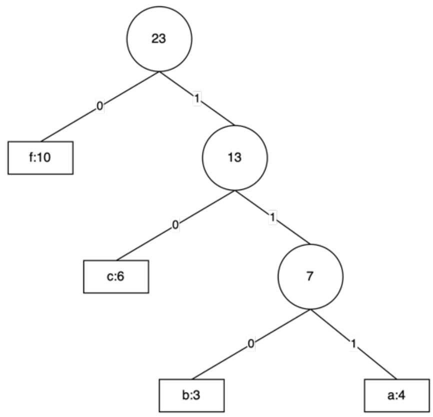
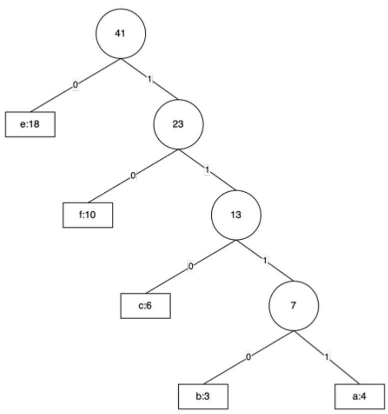
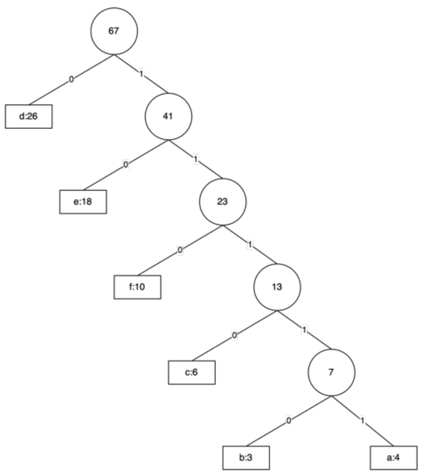
Como $f(n) = O\left(n^{\log_b a}\right)$, pois $n^{\log_b a} = n^{log_2 4} = n^2$.

Portanto, $n^{log_b a} = f(n)$, a solução é $T(n) = \Theta(n^2 \, log_2 n) = \Theta(f(n) \, log_2 n)$.

3. Monte os códigos e a árvore de Huffman codes para o seguinte dicionário (letra:frequência): {[a:4], [b:3], [c:6], [d:26], [e:18], [f:10]}.

```
HUFFMAN(S)
n = |S|
Q = S
for i = 1 to n − 1
    allocate a new node w
    y = EXTRACT-MIN(Q)
    z = EXTRACT-MIN(Q)
    w.left = z
    w.right = y
    w.freq = z.freq + y.freq
    INSERT(Q, w)
return EXTRACT-MIN(Q)    // the root of the tree is the only node left
```

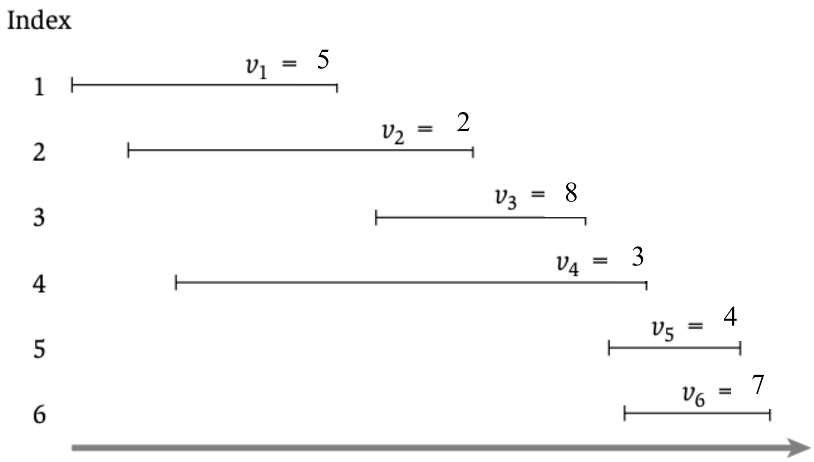
Resposta:

<table>
<tr>
<td>1) {[b:3] ,[a:4], [c:6], [f:10], [e:18], , [d:26]}</td>
<td>2) {[c:6], 7, [f:10], [e:18], [d:26]}

</td>
</tr>
<tr>
<td>3) {[f:10], 13, [e:18], [d:26]}

</td>
<td>4) { [e:18], 23, [d:26]}

</td>
</tr>
<tr>
<td>5) { [d:26], 41}

</td>
<td>5) {67}


d:0
e:10
f:110
c:1110
b:11110
a:11111</td>
</tr>
</table>

4. Utilize a abordagem de programação dinâmica com memoização para resolver a alocação de tarefas ponderadas abaixo. Calcule OPT(6) e indique quais requisições fazem parte da solução ótima dada por OPT(6).

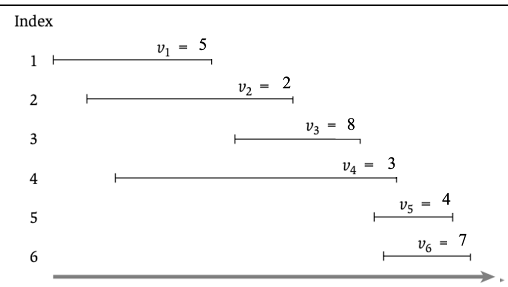
$$\boldsymbol{OPT(j)} = \begin{cases} \mathbf{0} & \boldsymbol{, se\ j = 0} \\ \boldsymbol{max\left(v_j + OPT(p(j)), OPT(j-1)\right)} & \boldsymbol{, se\ j > 0} \end{cases}$$



Resposta:

1) Computar os valores $p(j)$

```
M-Compute-Opt(j)
  If j=0 then
    Return 0
  Else if M[j] is not empty then
    Return M[j]
  Else
   Define M[j] = max(v_j+M-Compute-Opt(p(j)), M-Compute-Opt(j − 1))
    Return M[j]
 Endif
```



$p(1) = 0$

$p(2) = 0$

$p(3) = 1$

$p(4) = 0$

$p(5) = 3$

$p(6) = 3$

2) M-Compute-Opt(6)

$$\boldsymbol{OPT}(6) = \boldsymbol{M}[6] = \boldsymbol{max}\left(\boldsymbol{v_6} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{6})\big), \mathrm{M-Compute-Opt}(\mathbf{5})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_6} + \mathrm{M-Compute-Opt}(\mathbf{3}), \mathrm{M-Compute-Opt}(\mathbf{5})\big)$$

$$\boldsymbol{OPT}(5) = \boldsymbol{M}[5] = \boldsymbol{max}\left(\boldsymbol{v_5} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{5})\big), \mathrm{M-Compute-Opt}(\mathbf{4})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_5} + \mathrm{M-Compute-Opt}(\mathbf{3}), \mathrm{M-Compute-Opt}(\mathbf{4})\big)$$

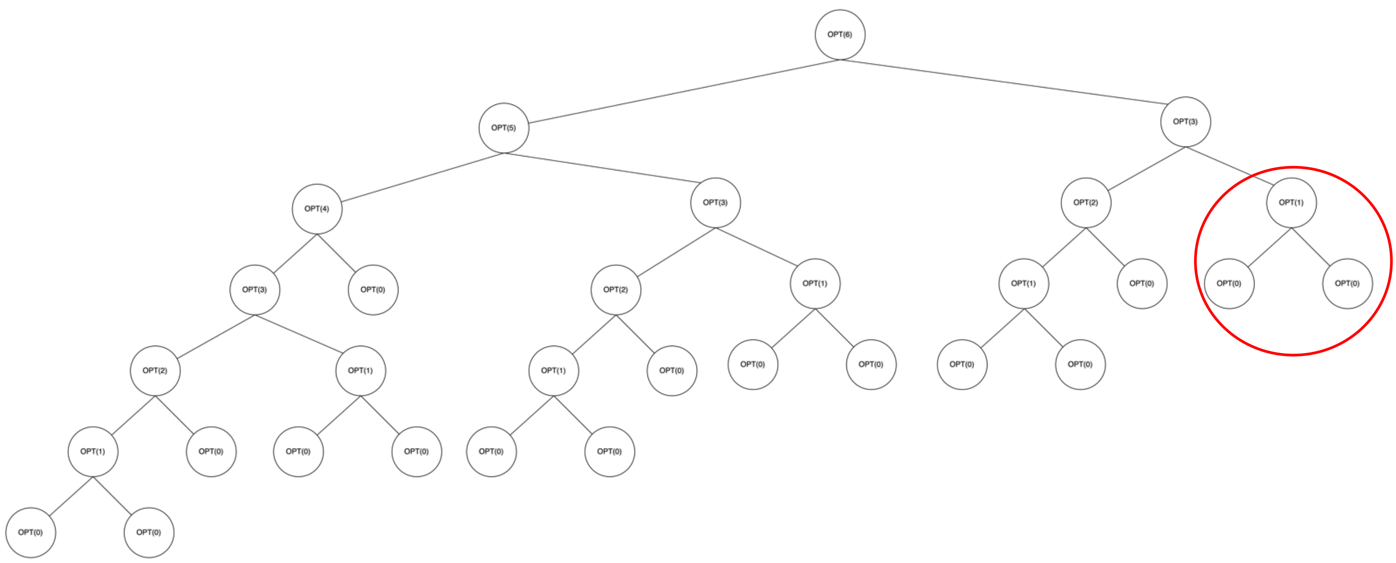
$$\boldsymbol{OPT}(4) = \boldsymbol{M}[4] = \boldsymbol{max}\left(\boldsymbol{v_4} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{4})\big), \mathrm{M-Compute-Opt}(\mathbf{3})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_4} + \mathrm{M-Compute-Opt}(\mathbf{0}), \mathrm{M-Compute-Opt}(\mathbf{3})\big)$$

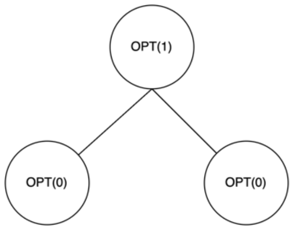
$$\boldsymbol{OPT}(3) = \boldsymbol{M}[3] = \boldsymbol{max}\left(\boldsymbol{v_3} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{3})\big), \mathrm{M-Compute-Opt}(\mathbf{2})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_3} + \mathrm{M-Compute-Opt}(\mathbf{1}), \mathrm{M-Compute-Opt}(\mathbf{2})\big)$$

$$\boldsymbol{OPT}(2) = \boldsymbol{M}[2] = \boldsymbol{max}\left(\boldsymbol{v_2} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{2})\big), \mathrm{M-Compute-Opt}(\mathbf{1})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_2} + \mathrm{M-Compute-Opt}(\mathbf{0}), \mathrm{M-Compute-Opt}(\mathbf{1})\big)$$

$$\boldsymbol{OPT}(1) = \boldsymbol{M}[1] = \boldsymbol{max}\left(\boldsymbol{v_1} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{1})\big), \mathrm{M-Compute-Opt}(\mathbf{0})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_1} + \mathrm{M-Compute-Opt}(\mathbf{0}), \mathrm{M-Compute-Opt}(\mathbf{0})\big)$$

$$\boldsymbol{OPT}(\mathbf{0}) = \boldsymbol{M}[\mathbf{0}] = \mathbf{0}$$



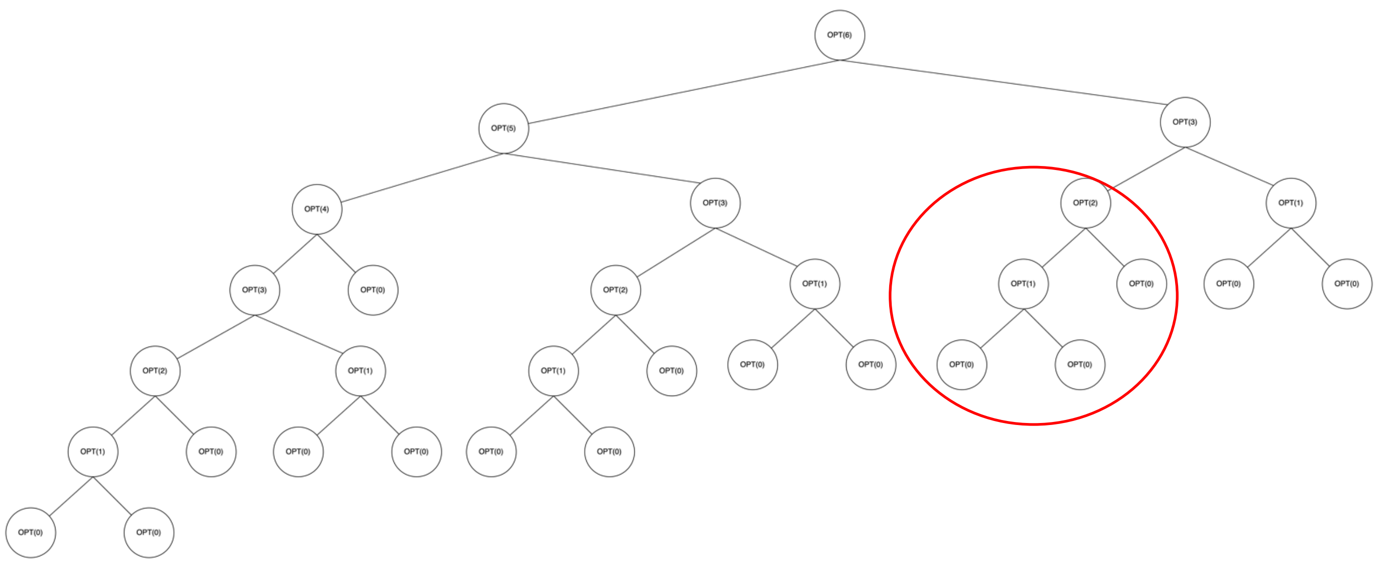


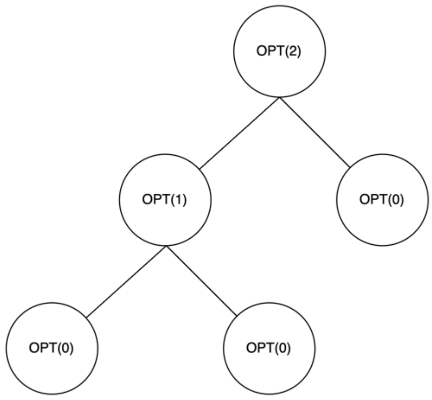
j	0	1	2	3	4	5	6
M[]	0						

$$\boldsymbol{OPT}(\mathbf{0}) = \boldsymbol{M}[\mathbf{0}] = \mathbf{0}$$

$$\begin{aligned} \boldsymbol{OPT}(\mathbf{1}) = \boldsymbol{M}[\mathbf{1}] &= \boldsymbol{max}\left(\boldsymbol{v_1} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{1})\big), \mathrm{M-Compute-Opt}(\mathbf{0})\right) \\ &= \boldsymbol{max}\big(\boldsymbol{v_1} + \mathrm{M-Compute-Opt}(\mathbf{0}), \mathrm{M-Compute-Opt}(\mathbf{0})\big) \\ &= \boldsymbol{max}(\boldsymbol{v_1} + \mathrm{M}[0], \mathrm{M}[0]) = \boldsymbol{max}(\mathbf{5} + 0, 0) = \mathbf{5} \end{aligned}$$

j	0	1	2	3	4	5	6
M[]	0	5					

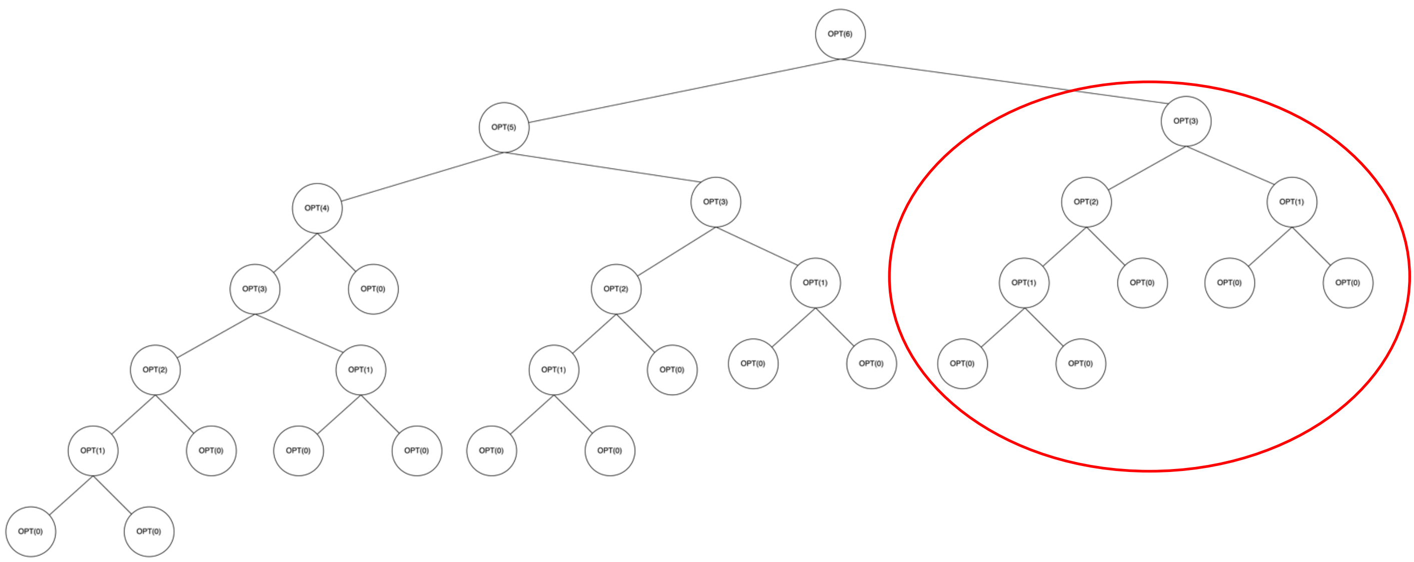


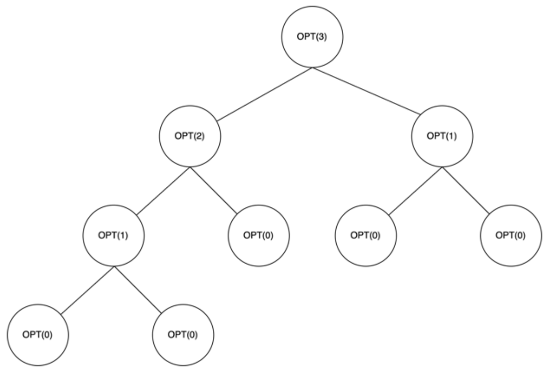


j	0	1	2	3	4	5	6
M[]	0	5					

$$\boldsymbol{OPT}(\mathbf{2}) = \boldsymbol{M}[\mathbf{2}] = \boldsymbol{max}\left(\boldsymbol{v_2} + \mathrm{M-Compute-Opt}\left(\boldsymbol{p}(\mathbf{2})\right), \mathrm{M-Compute-Opt}(\mathbf{1})\right)$$
$$= \boldsymbol{max}\left(\boldsymbol{v_2} + \mathrm{M-Compute-Opt}(\mathbf{0}), \mathrm{M-Compute-Opt}(\mathbf{1})\right)$$
$$= \boldsymbol{max}(\boldsymbol{v_2} + \mathrm{M}[\mathbf{0}], \mathrm{M}[\mathbf{1}]) = \boldsymbol{max}(\mathbf{2} + 0{,}5) = \mathbf{5}$$

j	0	1	2	3	4	5	6
M[]	0	5	5				

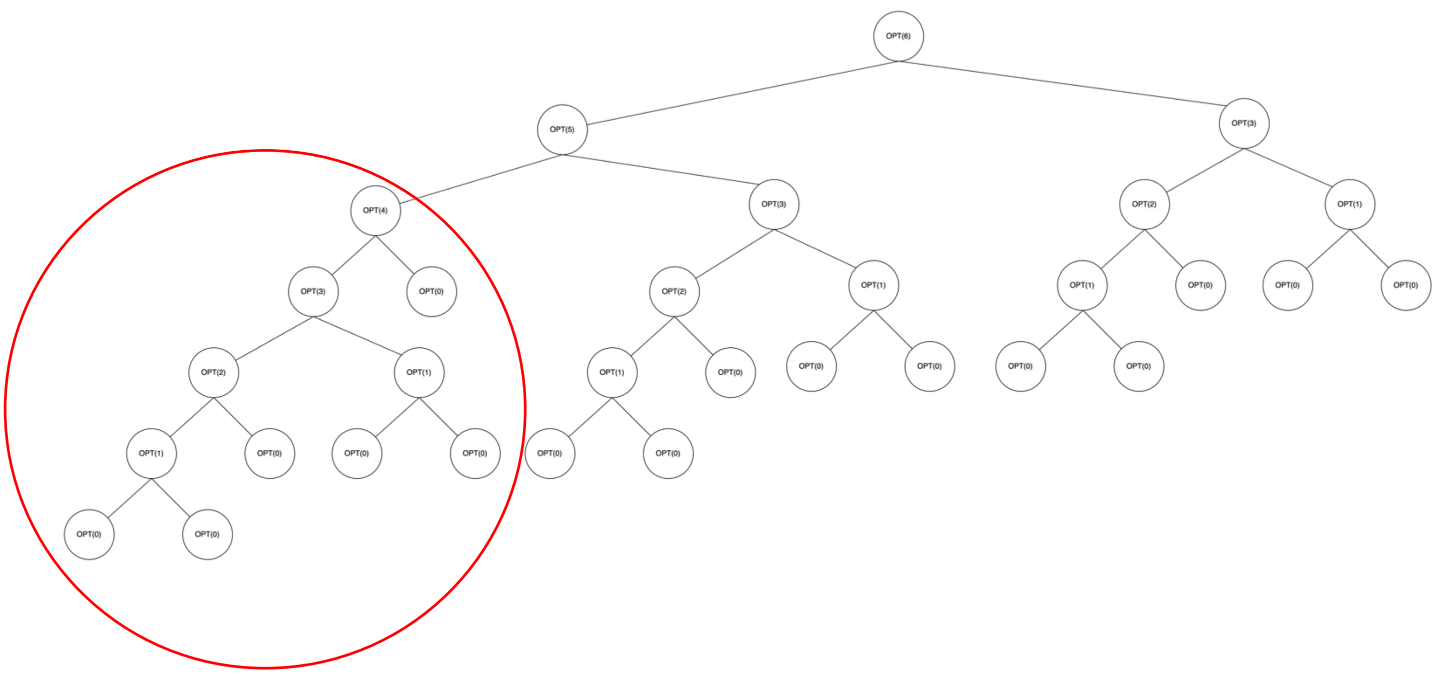


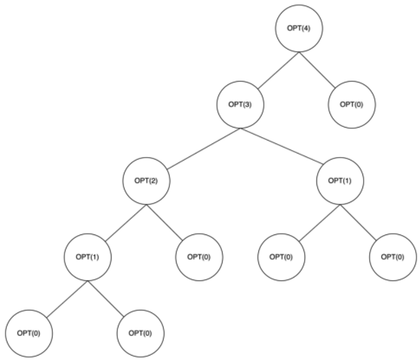


j	0	1	2	3	4	5	6
M[]	0	5	5				

$$\boldsymbol{OPT}(\mathbf{3}) = \boldsymbol{M}[3] = \boldsymbol{max}\left(\boldsymbol{v_3} + \mathrm{M-Compute-Opt}\left(\boldsymbol{p}(\mathbf{3})\right), \mathrm{M-Compute-Opt}(\mathbf{2})\right)$$
$$= \boldsymbol{max}\left(\boldsymbol{v_3} + \mathrm{M-Compute-Opt}(\mathbf{1}), \mathrm{M-Compute-Opt}(\mathbf{2})\right)$$
$$= \boldsymbol{max}(\boldsymbol{v_3} + \mathrm{M}[\mathbf{1}], \mathrm{M}[\mathbf{2}]) = \boldsymbol{max}(\mathbf{8} + 5{,}5) = \mathbf{13}$$

j	0	1	2	3	4	5	6
M[]	0	5	5	13			

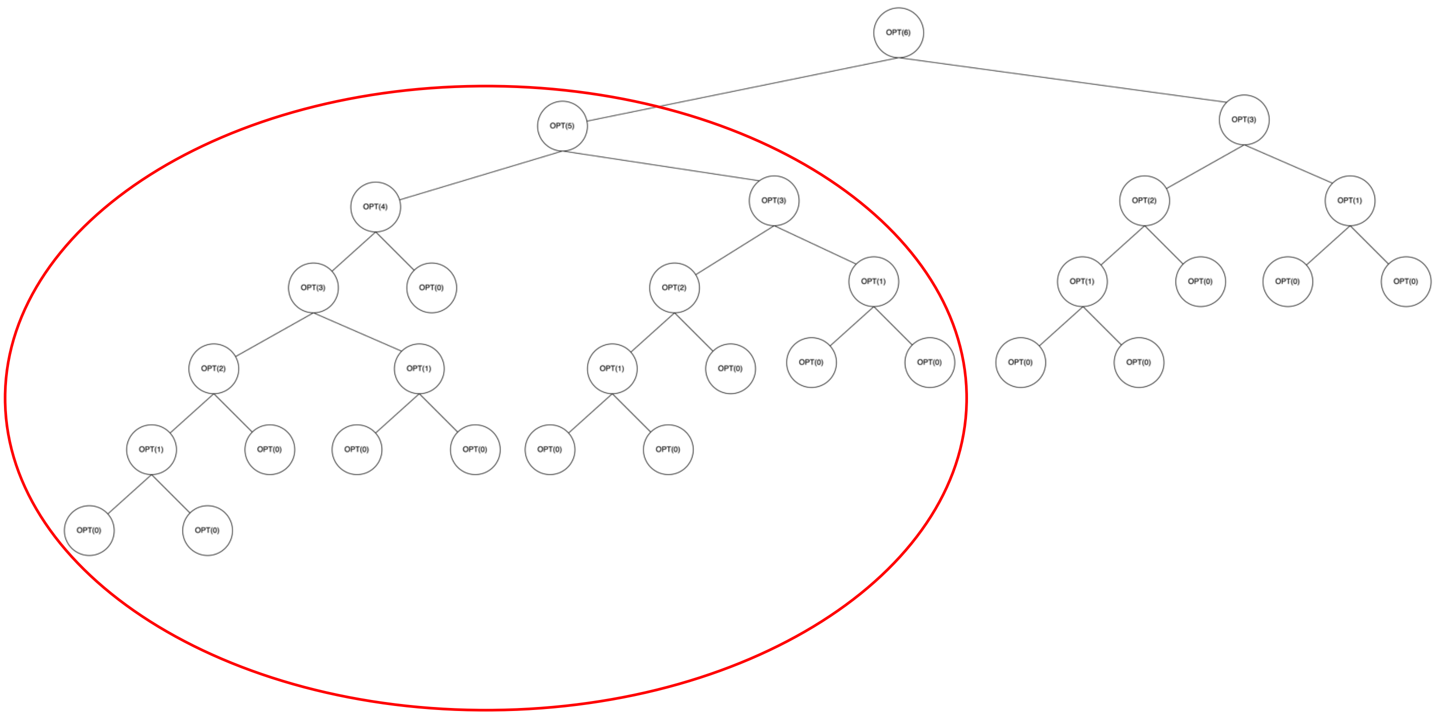


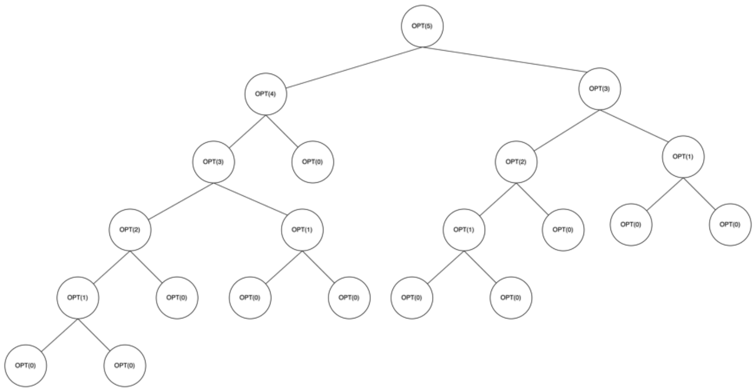


j	0	1	2	3	4	5	6
M[]	0	5	5	13			

$$\boldsymbol{OPT}(\mathbf{4}) = \boldsymbol{M}[\mathbf{4}] = \boldsymbol{max}\left(\boldsymbol{v_4} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{4})\big), \mathrm{M-Compute-Opt}(\mathbf{3})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v_4} + \mathrm{M-Compute-Opt}(\mathbf{0}), \mathrm{M-Compute-Opt}(\mathbf{3})\big)$$
$$= \boldsymbol{max}(\boldsymbol{v_4} + \mathrm{M}[\mathbf{0}], \mathrm{M}[\mathbf{3}]) = \boldsymbol{max}(\mathbf{3} + 0{,}13) = \mathbf{13}$$

j	0	1	2	3	4	5	6
M[]	0	5	5	13	13		





j	0	1	2	3	4	5	6
M[]	0	5	5	13	13		

$$\boldsymbol{OPT}(\mathbf{5}) = \boldsymbol{M}[\mathbf{5}] = \boldsymbol{max}\left(\boldsymbol{v}_{\mathbf{5}} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{5})\big), \mathrm{M-Compute-Opt}(\mathbf{4})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v}_{\mathbf{5}} + \mathrm{M-Compute-Opt}(\mathbf{3}), \mathrm{M-Compute-Opt}(\mathbf{4})\big)$$
$$= \boldsymbol{max}(\boldsymbol{v}_{\mathbf{5}} + \mathrm{M}[\mathbf{3}], \mathrm{M}[\mathbf{4}]) = \boldsymbol{max}(\mathbf{4} + 13{,}13) = \mathbf{17}$$

j	0	1	2	3	4	5	6
M[]	0	5	5	13	13	17	

$$\boldsymbol{OPT}(\mathbf{6}) = \boldsymbol{M}[\mathbf{6}] = \boldsymbol{max}\left(\boldsymbol{v}_{\mathbf{6}} + \mathrm{M-Compute-Opt}\big(\boldsymbol{p}(\mathbf{6})\big), \mathrm{M-Compute-Opt}(\mathbf{5})\right)$$
$$= \boldsymbol{max}\big(\boldsymbol{v}_{\mathbf{6}} + \mathrm{M-Compute-Opt}(\mathbf{3}), \mathrm{M-Compute-Opt}(\mathbf{5})\big)$$
$$= \boldsymbol{max}(\boldsymbol{v}_{\mathbf{6}} + \mathrm{M}[\mathbf{3}], \mathrm{M}[\mathbf{5}]) = \boldsymbol{max}(\mathbf{7} + 13{,}17) = \mathbf{20}$$

j	0	1	2	3	4	5	6
M[]	0	5	5	13	13	17	20

3) Valores que compõem a solução ótima M-Compute-Opt(6)

FIND-SOLUTION(j)

IF ($j = 0$)
  RETURN $\varnothing$.
ELSE IF ($w_j + M[p[j]] > M[j-1]$)
  RETURN $\{ j \} \cup$ FIND-SOLUTION($p[j]$).
ELSE
  RETURN FIND-SOLUTION($j-1$).

$$\boldsymbol{S} = \{\quad\}$$

$\boldsymbol{Find-Solution}(\mathbf{6}) = M[6] \geq M[5]$	$\boldsymbol{S} = \{\mathbf{6}\}$
$\boldsymbol{Find-Solution}(\boldsymbol{p}(\mathbf{6})) = M[3] \geq M[2]$	$\boldsymbol{S} = \{\mathbf{3}, \mathbf{6}\}$
$\boldsymbol{Find-Solution}(\boldsymbol{p}(\mathbf{3})) = M[1] \geq M[0]$	$\boldsymbol{S} = \{\mathbf{1}, \mathbf{3}, \mathbf{6}\}$